A DICTIONARY APPROACH TO REPETITIVE PATTERN FINDING IN MUSIC

Hsuan-Huei Shih, Shrikanth S. Narayanan and C.-C. Jay Kuo
Integrated Media Systems Center and Department of Electrical Engineering-Systems
University of Southern California, Los Angeles, CA 90089-2564
Tel: (213) 740-8386, Fax: (213) 740-4651, E-mail: {hshih,shri,cckuo}@sipi.usc.edu

ABSTRACT

A dictionary-based approach for extracting repetitive patterns in music aiming at music feature extraction and indexing for audio database management is proposed. In this system, segmentation is achieved with the tempo information, and a music score is decomposed into bars. Each bar is indexed to construct a bar index table. Then, an adaptive dictionary-based compression algorithm known as Lempel Ziv 78 (LZ-78) is applied to the bar-represented music scores to extract repetitive patterns. Finally, pruning is applied to this dictionary to remove non-repeating patterns and to combine shorter repeating patterns into a longer one. The LZ78 algorithm is slightly modified to achieve better results in the current application context. Experiments performed on a popular music database of MIDI files demonstrated that the proposed algorithm extracts repeating melodies effectively with a speed of four times faster compared to the traditional linear search approach.

1. INTRODUCTION

Techniques for image and video feature extraction, indexing and retrieval have received a lot of attention recently in image and video database applications. In contrast, a relatively small amount of effort has been put into audio feature extraction and audio database indexing. Audio database management finds applications in music archiving, special effect sound search in audio editing, etc. Audio is also an integral part of multimedia databases, and often contains useful information for effective multimedia search. Multimedia database management using multimodal information such as audio, video and text, is an emerging trend. A better understanding of audio features and their utilization is an essential step towards creating a complete multimedia database management system. Within the context of audio databases, music databases have received considerably less attention. However, music is a universal concept and language, and study on how people understand and interact with music is important.

Some work has been done in music content analysis and database organization before. Chen *et al.* [1] proposed a pat-tree approach to index melodies while Ghias *et al.* [2] used coarse melody contours as a key to query a music database. McNab, *et al.* [7],[8] used interval contours for interactive music retrieval. Tong and Kuo [9] considered a hidden Markov model (HMM) method to model special effect sounds for content-based audio query. Furthermore, Chen, *et al.* proposed the string-join approach [3] and the correlative matrix approach [4] to find repeating patterns in music. Both these approaches use notes as their basic units. However, the computational complexity grows rapidly as the number of notes increases. Moreover, the essential duration information of each note was discarded in these systems. In contrast, our proposed system uses bars instead of notes as the basic unit. It not only captures tempo information of melodies but can also reduce the size of the input sequence.

In this work, we focus on the extraction of repeating patterns corresponding to the main melody of a given music piece. Repeating patterns can be used in organizing and indexing music databases. They can also serve as an important feature for content-based retrieval from music databases. Furthermore, they can be used as a tool for analyzing characteristics and patterns of music compositions and their composers. It is believed that people are particularly sensitive and receptive to certain salient portions in a piece of music. Here, we assume that repeating melodies constitute such a salient part. It is not uncommon that a piece of music is composed by certain small pieces of melody that tend to be repeated throughout the whole piece. Therefore, people tend to easily memorize repeating melodies. If a piece of music is written in music score form, repeating melodies in a piece of music are repeating patterns of notes in its music score. We use a dictionary approach to find these repeating patterns based on the classic work of Lempel and Ziv [5],[6].

The rest of this paper is organized as follows. In Section 2, the proposed algorithm is described in detail. Experimental results are given in Section 3 and concluding remarks are presented in Section 4.

2. PROPOSED ALGORITHM

The input to our proposed system is a piece of music consisting of numerical music scores. Thus, it is assumed that other music forms such as the sound waves and MIDI files have been first converted to numerical music scores.

2.1 System Overview

Figure 1 is the functional flow diagram of the overall system. The two main phases in the system are: data preparation and repeating pattern extraction. Music decomposition and bar indexing constitute the data preparation phase. The two main modules in the repeating pattern extraction phase include modified LZ78 processing and dictionary pruning. The extraction of repeating patterns is done iteratively. A repeating pattern list is introduced to store the extracted full-length repeating patterns. A repeating pattern is said to be of full-length if it is not a proper subset of any other repeating pattern that has the same frequency count. A dictionary is generated after each LZ78 iteration and pruned to remove non-repeating patterns. Moreover, the extracted full-length repeating patterns are moved into a repeating pattern list.

The pruned dictionary is passed on to the next LZ78 iteration. The iteration is terminated when the system converges i.e., when pruned dictionaries of the current and previous iterations are the same.

Further details of each of the modules mentioned above are described in the following sections.

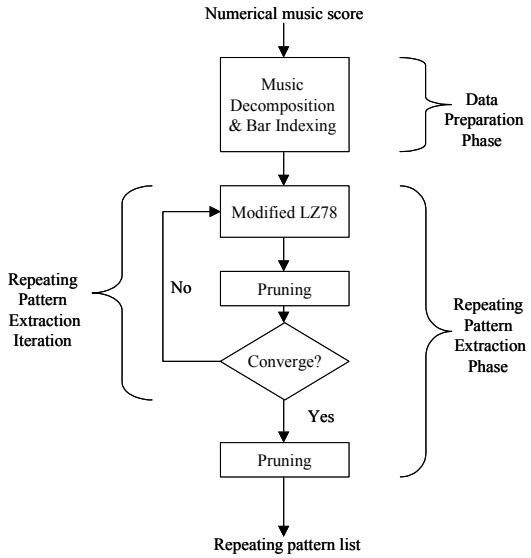

Figure 1. System block diagram

2.2 Music Decomposition and Bar Indexing

The bar (rather than the note) is used as a basic unit in our system due to the following considerations. First, a music note is too fine a unit to build a dictionary with since there are too many notes, and their combinations, in a piece of music and the complexity of the dictionary building process grows very rapidly. Second, a note contains pitch and duration information. However, if a note is used as a symbol to build a dictionary, the duration of a note is often discarded for simplification. Here, bars are introduced to preserve the duration information of notes. In music scores, there are time signatures used to indicate the tempo of the underlying music, and a single piece of music may contain more than one time signature. In a music score, bars are used to group notes together according to a specified time signature. In our algorithm, bars are chosen to be the basic unit where a group of notes of the same time period are cascaded.

Usually, several bars form a repeating pattern. However, a repeating pattern may not start precisely at the beginning of a bar or stop at the end of a bar. In other words, they may start or stop at any note in a bar. For a given song, repeating patterns tend to start and stop at fixed positions in a bar. The intermediate bars that lie between the starting and the ending bars are exactly the same. Just the leading and trailing bars of a repeating pattern require some special handling.

When the bar index table is built, the segmented music score is also concurrently converted into a bar indexed music score. This merely implies replacing each bar in the music score with its corresponding index. Furthermore, we can record pitch values of consecutive notes in a bar while ignoring their durations and, at the same time, discard all rest notes to derive another attribute. Rests at the start, in the middle, or at the end of a bar are treated the same in the bar matching process. By making this assumption several different bars which have same number of notes with the same pitch values will be matched to the same index in the bar index table. However, it should be noted that such combinations of notes occur rarely in the same piece of music. A non-trivial repeating pattern is a sequence of several bars. A single bar usually does not contribute toward discrimination since it is quite difficult for people to identify a particular piece after listening to only one bar. Therefore, one bar is too short to be considered as a repeating pattern.

2.3 Lempel-Ziv 78 (LZ78) and Its Modification

The Lempel-Ziv 78 (LZ78) algorithm is a lossless compression scheme that has been widely used in text compression. A dictionary of variable length is constructed and adaptively updated by LZ78 while parsing a sequence of symbols. Vocabularies in the dictionary will be added according to the processed data. In our system, input symbols are bars with an appropriate index number and vocabularies in the dictionary are sequences of bar indices. Sequences of bar indices are called patterns through out this paper. The main idea of dictionary-based compression is to detect longer vocabulary entries and encode them with shorter codewords. This process turns out to be a powerful tool in finding repeating patterns in music. The dictionary is the place where repeating patterns are accumulated.

"DA" is called the parent pattern of both "DAD" and "DAB". In general, to form a parent pattern that is N bars long by using LZ78 requires that the pattern appears at least N times in the underlying music. If N is large, it could be difficult to get a long parent pattern. A long parent pattern is needed for longer repeating patterns. To overcome this difficulty, we pass the same music piece through the LZ78 dictionary building system several times, and the dictionary in each LZ78 iteration is built based on the previously built dictionary.

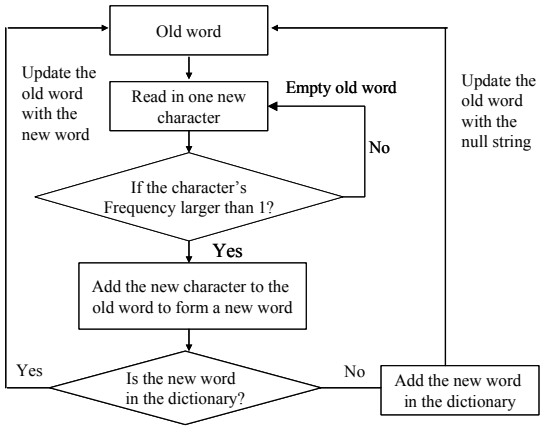

Figure 2. The block diagram of modified LZ78

The flow diagram of the modified LZ78 (MLZ78) is shown in Figure 2, and explained below.

1. A buffer and a dictionary are needed in MLZ78, and at the beginning they are both empty. The buffer is referred to as old/new word.

2. One new character is read in from the incoming data. If the bar index frequency is 1, empty the old/new word buffer and start from Step 2 again. Otherwise, go to the next step.
3. Append the new character to the old word, and it becomes the new word. There is at least one character in the new word buffer.
4. There are two cases. (a) If the new word is already in the dictionary, then this new word becomes the old word (and nothing is changed in the buffer). Start from Step 2 again. (b) If the new word is not in the dictionary, add the new word to the dictionary, empty the buffer and return an empty old word. Then, record the index of newly added pattern's parent. Start from Step 2 again.

Patterns in the dictionary may not be repeating patterns, and furthermore all parents of patterns are also included in the dictionary. The dictionary will diverge if these non-repeating patterns and pattern's parents are not handled properly. Hence, pruning the dictionary after each modified LZ78 iteration is essential to have a convergent dictionary, thus enabling easier extraction of repeating patterns. Details of the pruning techniques are discussed in the next section.

2.4 Pruning

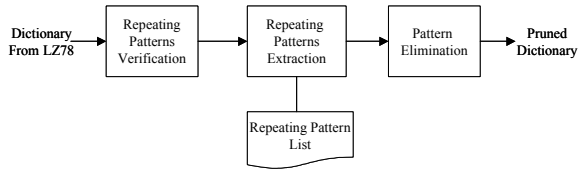

Figure 3. Flow diagram of the pruning phase

Figure 3 shows a flow diagram of the pruning algorithm. The pruning phase has three stages: repeating pattern verification, repeating pattern extraction, and pattern elimination. In the first stage, repeating pattern verification, the repetition of each pattern is verified by using the bar indexed music score. Although the proposed modified LZ78 tries to eliminate the problem of having non-repeating patterns in the dictionary, some non-repeating patterns may still appear in the dictionary. The main reason for this phenomenon is a bar index that is not a part of any repeating patterns may have multiple appearances in a piece of music. These appearances typically occur in isolated places and the modified LZ78 algorithm has no way of detecting this problem. Since the dictionary converges, the time required for checking repetitions will not grow as the number of modified LZ78 iterations increases. All patterns in the dictionary will be checked for their repetition, while at the same time the frequency attribute in the dictionary will be updated.

In the repeating pattern extraction stage, full-length repeating patterns will be extracted and moved from the dictionary to a repeating pattern list. A subset of a full-length repeating pattern may have a higher frequency than the full-length repeating pattern. Then, this subset may be another full-length repeating pattern if it is not a proper subset of other repeating patterns that have the same frequencies. All entries in the repeating pattern list are full-length repeating patterns. A threshold will be set to tell the system what the minimum length of a pattern should be for it to be considered as a repeating pattern. There are two ways in arriving at repeating patterns. One is by detecting non-repeating patterns, and the other is finding patterns that stop growing after several modified LZ78 iterations. For the first case, a pattern of length N is not a repeating pattern but its parent patterns of length N-1 must be repeating patterns. Otherwise, the non-repeating patterns will not be able to extend to length N. Since the dictionary keeps track of each pattern's parent. Then, extracting a full-length repeating pattern from the dictionary becomes straightforward. For the second case, since repeating patterns are terminated by indices that have frequencies equal to 1, these patterns will not grow any longer, and can be left until the end to be extracted. Therefore, a final pruning step will be applied right before the system returns a repeating pattern list.

In the pattern elimination stage, non-repeating patterns are removed from the dictionary. Patterns of length equal to one are also removed. Extracted repeating patterns are also removed. Moreover, proper subsets of an extracted full-length repeating pattern will be eliminated as well. Since some patterns are removed from the dictionary, the indices of patterns will be reordered in the pruned dictionary. The consistency of parents' indexes will be updated in the pruned dictionary. Then, this pruned dictionary will be used in the next MLZ78 iteration, until the pruned dictionary stops changing with respect to the previous pruned dictionary.

3. EXPERIMENTAL RESULTS

3.1 Experiment Setup and Example

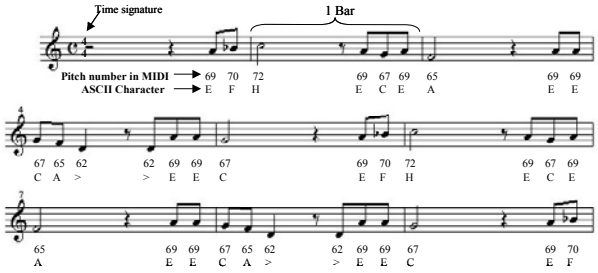

Figure 4. The First nine bars of the music score of "Yellow Submarine"

We collected 200 MIDI files of the seventies' and eighties' pop music genre from the public domain to form our database. The system was implemented in VC++ on an Intel PC. According to the time signature information in the MIDI file, we segmented numerical music scores into bars and then applied our algorithm to the bar representation. For illustration, we will use the piece "Yellow Submarine" by Beatles as an example. Figure 4 shows the first nine bars of Yellow Submarine in music scores and the numerical pitch values are specified under each note (the pitch values are obtained from the MIDI file). After converting the music score to the numerical music score, it is segmented into bars according to the time signature, which is 4/4 for Yellow Submarine. Then, the decomposed numerical music score is used to build a bar index table and converted into bar indexed music score based on the bar index table.

The left part of Figure 5 shows the entire bar indexed music score. We applied the modified LZ78 to the bar indexed music score and pruned the dictionary over several iterations until the dictionary finally converged. The dictionary of this example converged after 15 iterations. The threshold for the minimum length of a repeating pattern was set to 3. Finally, the algorithm extracted 5 repeating patterns as shown in the right part of Figure 5.

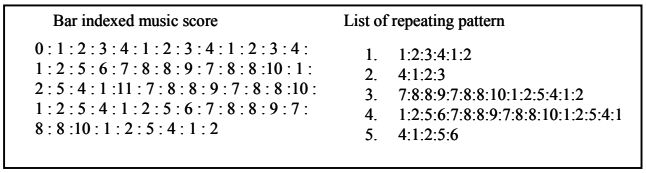

Figure 5. Bar index music score of Yellow Submarine (left) and the final result of repeating patterns in Yellow Submarine (right).

3.2 Experimental Result Analysis

Title	NO. of bars	Bar index table Size	NO. of MLZ 78 iterations	NO. of repeating patterns	Max. repeating pattern Length
Yellow Submarine	59	12	15	5	17
All I have to do is dream	55	27	4	3	5
500 miles	52	24	6	4	5
Hotel California	117	22	21	2	15

Table 1. Statistical information of tested songs

The number of extracted repeating patterns was found to vary across different songs. Some songs tended to have more repeating patterns than others. The length of repeating patterns also varied across songs. The number of segmented bars of a piece of music is dependent of the length of the input music. However, the bar index table of a piece of music will be dependent on the nature of the piece of music. In our system, we only set the minimum length threshold since very short repeating patterns were not deemed to be important. Some non-trivial (i.e., lengthy) repeating patterns may be extracted from a music piece. However, repeating patterns that may be embedded within a full-length repeating pattern are difficult to be extracted. In our example, in Figure 5 on the right, the pattern "7:8:8" happened two times in pattern number 3. In fact, a repeating pattern in a full-length repeating pattern may or may not sound as a complete repeating melody, since only certain combinations of notes can be used to end a melody. Therefore, the proposed system is used only to extract full-length repeating patterns. Table 1 shows the statistical information of tested songs. As seen in Table 1, some repeating patterns end up being very long after pruning. For example, Hotel California by Eagle has only two very long repeating patterns that are truly the two main melodies as confirmed by informal listening.

4. CONCLUSION AND FUTURE WORK

A dictionary-based approach was developed to find repeating patterns for music feature extraction and indexing. It was shown with experimental results that the proposed method could detect repeating patterns in music effectively.

In the future, we would like to continue our work on efficient pruning techniques for the proposed dictionary approach to enhance obtained results. Further improvement of the modified Lempel Ziv 78 (MLZ78) will also be carried out to give a better intermediate result for pruning. Also, since MIDI files are used as input to our system, the bar representation used for pattern extraction is unambiguous. However, when a piece of music is either played or sung by people, we have to convert the acoustic waveform to the bar representation in a preprocessing step. This demands robust signal processing techniques. Besides, since the bar representation may not be as accurate as that obtained from MIDI files, we have to develop a matching process that permits a certain level of error tolerance. This may require statistical approaches to music pattern extraction.

5. REFERENCES

[1] A. L. P. Chen, and C. C. Liu, "Music databases: indexing techniques and implementation", *Proceedings IEEE Intl. Workshop on Multimedia Data Base Management Systems*.1999.

[2] A. Ghias, J. Logan, D. Chamberlin, and B. C. Smith. "Query by humming: musical information retrieval in an audio database", *Proceedings of ACM Multimedia Conference'95*, San Francisco, California, November 1995.

[3] C. C. Liu, J. L. Hsu and A. L. P. Chen, "Efficient theme and non-trivial repeating pattern discovering in music databases," *Proc. IEEE International Conference on Data Engineering.* 1999.

[4] J. L. Hsu, C. C. Liu and A. L. P. Chen, "Efficient repeating pattern finding in music databases," *Proc. ACM Seventh International Conference on Information and Knowledge Management (CIKM).* 1998.

[5] J. Ziv and A. Lempel,. "A universal algorithm for sequential data compression," *IEEE Transactions on Information Theory,* Volume 23, Number 3, September 1977, pp. 337-343.

[6] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," *IEEE Transactions on Information Theory,* Volume 24, Number 5, September 1978, pp. 530-536.

[7] R. J. McNab, *Interactive Applications of Music Transcription,* Master's thesis, Department of Computer Science, University of Waikato, New Zealand, 1996.

[8] R. J. McNab, L. A. Smith, I. H. Witten, C. L. Henderson, and S. J. Cunningham, "Towards the digital music library: Tune retrieval from acoustic input," In *Digital Libraries Conference*, 1996.

[9] T. Zhang and C.-C. J. Kuo, Content-based Audio Classification and Retrieval for Audiovisual Data Parsing, Kluwer Academic Publishers, 2001